INFORMATION AND CONTROL 18, 311–318 (1971)

Cap Expressions for Context-Free Languages

M. K. YNTEMA

The Pennsylvania State University, University Park, Pennsylvania 16802

A new operation on languages, called the "cap operation" is introduced, and corresponding "cap expressions" are defined. It is shown that concatenation, union, and cap are sufficient to characterize context free languages. Furthermore there are effective procedures for finding a cap expression from a context-free grammar and vice versa.

Regular expressions characterize regular sets, i.e., a set is regular if and only if there is a regular expression denoting it. An attempt was made by Yntema (1965, 1967) to extend this notation to include the context-free languages by introducing a matching choice operation. The result was to characterize a subfamily of the context-free languages, the standard-matching-choice sets. These are the superlinear languages of Brzozowski (1968), and the derivation-bounded languages of Ginsburg and Spanier (1968). In this paper a new operation, called the cap operation, is introduced which, along with union and concatenation, characterizes context-free languages. Actually the cap operator can be used with any notation that permits substitution of languages for letters. It will be shown that any such notation for subfamilies of context free languages which include all of the finite sets can be combined with the cap operator to characterize context free languages.

DEFINITION. Let $P(\alpha)$ be an expression for a set of words over an alphabet containing α, such that $P(\alpha)$ does not contain the symbol $\hat{\alpha}$. $\langle P(\hat{\alpha})\rangle_\alpha$ represents the set of all words which do not contain α, but which can be obtained from $P(\alpha)$ by a finite succession of substitutions of words of $P(\alpha)$ for α into words of $P(\alpha)$ and into words resulting from such substitutions. An expression $\langle P(\hat{\alpha})\rangle_\alpha$ will be called an *α-cap expression.* A *cap expression* is an α-cap expression for some letter α.

It should be noted that if $x \in P(\alpha)$ and x does not contain an occurrence of α, then $x \in \langle P(\hat{\alpha})\rangle_\alpha$. If β or $\hat{\beta}$ do not appear in the expression $P(\alpha)$, then $\langle P(\hat{\alpha})\rangle_\alpha = \langle P(\hat{\beta})\rangle_\beta$, since the words of the set denoted by a cap expression

do not contain the capped letter. So the capped letter becomes a bound variable.

Marking the capped letter within the expression and using it as an index outside the brackets seems redundant. Since there may be cap expressions within cap expressions the index marks which brackets are associated with which letters. The cap on the letter is a visual reminder that the variable is bound, no longer a simple letter of the alphabet.

LEMMA 1. *Let $P(\alpha)$ be an expression for a set on an alphabet containing α. Let $\mu_0\alpha\mu_1\alpha\cdots\mu_{n-1}\alpha\mu_n$ be a word of $P(\alpha)$ such that for each i μ_i does not have an occurrence of α. Let $y_i \in \langle P(\hat{\alpha})\rangle_\alpha$ $1 \leqslant i \leqslant n$. Then $\mu_0 y_1\mu_1\cdots\mu_{n-1}y_n\mu_n$ is a word of $\langle P(\hat{\alpha})\rangle_\alpha$.*

Proof. Each y_i can be derived by a finite succession of substitutions of words of $P(\alpha)$ into $P(\alpha)$ and words resulting from such substitutions. $\mu_0 y_1\mu_1 y_2\cdots\mu_{n-1}y_n\mu_n$ can be obtained by first substituting, for each i, the initial word of $P(\alpha)$ in the succession for deriving y_i for the appropriate α in $\mu_0\alpha\mu_1\alpha\cdots\mu_n$, and then continuing with the substitutions of the derivation of y_i.

LEMMA 2. *Let G be a context free grammar with initial variable α, and let G' be the grammar derived from G by replacing every occurrence of α in the right side of a production by a new letter α'. If $P(\alpha')$ is an expression for $L(G')$, then $\langle P(\hat{\alpha})\rangle_\alpha$ is an expression for $L(G)$.*

Proof. $x \in L(G)$ if and only if $\alpha \Rightarrow x$ and x is a word on the terminal alphabet. Assume $x \in L(G)$. The productions used in the generation $\alpha \Rightarrow x$ can be chosen in such a way that no substitution is made for α unless there is no other variable to use, and a substitution for α is always for the leftmost occurrence of α. Use induction on the length of the generation to show $x \in \langle P(\hat{\alpha})\rangle_\alpha$.

If the generation is of length 1, then the single production used is $\alpha \to x$, which belongs to both G and G'. Hence $x \in L(G') = P(\alpha')$. Since α' is not a letter in x, $x \in \langle P(\hat{\alpha}')\rangle_{\alpha'}$ which is the same as $\langle P(\hat{\alpha})\rangle_\alpha$.

If the length of the generation $\alpha \Rightarrow x$ is greater than 1, there are two possibilities. Either a production with α on the left is used after the initial production, or not. If not, by reasoning similar to that for a generation of length 1, $x \in \langle P(\hat{\alpha})\rangle_\alpha$.

Assume at least one substitution is made for α, after the initial production. Then the generation has the form:

$$\alpha \Rightarrow \mu_0\alpha\mu_1\alpha\cdots\alpha\mu_n \Rightarrow \mu_0 y_1\mu_1 y_2\cdots y_n\mu_n = x,$$

where, for each i, $\alpha \Rightarrow y_i$ and each such generation is shorter than the generation of $\alpha \Rightarrow x$. So, by the induction hypothesis $y_i \in \langle P(\hat{\alpha})\rangle_\alpha$.

Since $\mu_0\alpha\mu_1\alpha \cdots \alpha\mu_n$ is derived without any substitutions for α it belongs to $P(\alpha)$. x is derived from this word of $P(\alpha)$ by substituting words of $\langle P(\hat{\alpha})\rangle_\alpha$ for α. Hence $x \in \langle P(\hat{\alpha})\rangle_\alpha$ by Lemma 1.

Next, assume $x \in \langle P(\hat{\alpha})\rangle_\alpha$ and show it must belong to $L(G)$. Since x is in $\langle P(\hat{\alpha})\rangle_\alpha$, it is derived from $P(\alpha)$ by successive substitutions of words of $P(\alpha)$ for α, and x does not contain an occurrence of α. Use induction on the number of substitutions leading to words without α. If there are no substitutions, then $x \in P(\alpha)$ and $\alpha \Rightarrow x$. If there are substitutions then $x = \mu_0 y_1 \mu_1 y_2 \cdots y_n \mu_n$ where each $y_i \in \langle P(\hat{\alpha})\rangle_\alpha$, each μ_i has no occurrence of α, and $\mu_0\alpha\mu_1\alpha \cdots \alpha\mu_n \in P(\alpha)$. By the induction hypothesis $\alpha \Rightarrow y_i$, for each i. Hence

$$\alpha \Rightarrow \mu_0\alpha\mu_1\alpha \cdots \alpha\mu_n \Rightarrow \mu_0 y_1 \mu_1 y_2 \cdots y_n \mu_n = x$$

and $x \in L(G)$.

LEMMA 3. *If $P(\alpha)$ is an expression for a context free language, then $\langle P(\hat{\alpha})\rangle_\alpha$ also denotes a context free language.*

Proof. Let α' be a new letter. Since $P(\alpha)$ is context free, so is $P(\alpha')$. Let G' be a grammar for $P(\alpha')$ with α as initial variable. Let G be the grammar derived from G' by changing all occurrences of α' to α. By Lemma 2, $L(G) = \langle P(\hat{\alpha})\rangle_\alpha$.

A second proof can be given by starting with G as a grammar for $P(\alpha)$ with initial variable σ, then defining G_α as the grammar with initial variable α which has all of the productions of G plus the one additional production $\alpha \to \sigma$. Then $L(G_\alpha) = \langle P(\hat{\alpha})\rangle_\alpha$.

LEMMA 4. *Let G be a context free grammar with variable α and initial variable σ. Let G_α be the grammar with productions the same as G, but α as initial variable. Let G' and G_α' be the grammars derived from G and G_α, respectively, by replacing every occurrence of α in the right sides of productions by α', a new letter. If $P(\alpha') = L(G')$ and $Q(\alpha') = L(G_\alpha')$, then $P(\langle Q(\hat{\alpha})\rangle_\alpha) = L(G)$.*

Proof. By definition of $P(\alpha')$ and G_α, $L(G) = P(L(G_\alpha))$. By Lemma 2, $L(G_\alpha) = \langle Q(\hat{\alpha})\rangle_\alpha$. Hence $L(G_\alpha) = P(\langle Q(\hat{\alpha})\rangle_\alpha)$.

LEMMA 5. *Let G be a context free grammar with variables α and β, and initial variable σ. Let G_α and G_β be the grammars with the same productions as G but α and β as initial variables, respectively. Let G'', G_α'', and G_β'' be the*

grammars derived from G, G_α, *and* G_β, *respectively, by replacing every occurrence of* α *on the right side of a production by* α', *and every such occurrence of* β *by* β', α' *and* β' *being new letters. Let* $P(\alpha', \beta')$, $Q(\alpha', \beta')$, *and* $R(\alpha', \beta')$ *be expressions for* $L(G'')$, $L(G''_\alpha)$, *and* $L(G''_\beta)$, *respectively*:

(i) $L(G) = P(L(G_\alpha), L(G_\beta))$;
(ii) $L(G_\alpha) = \langle Q(\hat{\alpha}, \langle R(\hat{\alpha}, \hat{\beta})\rangle_\beta\rangle_\alpha$;
(iii) $L(G_\beta) = \langle R(\langle Q(\hat{\alpha}, \hat{\beta})\rangle_\alpha, \hat{\beta})\rangle_\beta$;
(iv) $L(G_\alpha) = \langle Q(\hat{\alpha}, \langle R(\langle Q(\hat{\gamma}, \hat{\beta})\rangle_\gamma, \hat{\beta})\rangle_\beta)\rangle_\alpha$;
(v) $L(G_\beta) = \langle R(\langle Q(\hat{\alpha}, \langle R(\hat{\alpha}, \hat{\delta})\rangle_\delta)\rangle_\alpha, \hat{\beta})\rangle_\beta$.

Proof. Part (i) follows immediately from the definitions. To show part (ii), let G_α' and G_β' be the grammars derived from G_α and G_β, respectively, by replacing every occurrence of α on the right side of a production by α'. Since $L(G''_\alpha) = Q(\alpha', \beta')$ and $L(G''_\beta) = R(\alpha', \beta')$, it follows by Lemma 4 that $L(G_\beta') = \langle R(\alpha', \hat{\beta})\rangle_\beta$ and $L(G_\alpha') = Q(\alpha', \langle R(\alpha', \hat{\beta})\rangle_\beta)$. Hence, by Lemma 2, $L(G_\alpha) = \langle Q(\hat{\alpha}, \langle R(\hat{\alpha}, \hat{\beta})\rangle_\beta)\rangle_\alpha$. The proof of part (iii) is similar to that of part (ii).

To show part (v), use the equation $L(G_\beta') = \langle R(\alpha', \hat{\beta})\rangle_\beta$, derived in the proof of part (ii). This yields $L(G_\beta) = \langle R(L(G_\alpha), \hat{\beta})\rangle_\beta$. Change the bound variable $\hat{\beta}$ in part (ii) to a new variable $\hat{\delta}$, for an expression for $L(G_\alpha)$. Substituting for $L(G_\alpha)$ in the above β-cap expression gives part (v). Without the change of variable the resulting expression would have violated the definition of a cap expression by having a β-cap expression within a β-cap expression.

THEOREM 1. *If* G *is a context free grammar, then* $L(G)$ *is expressible using concatenation, union, and cap.*

Proof. The proof is by induction on the number of variables in G. First, assume G has only one variable α, and let G' be the grammar derived from G by replacing every occurrence of α on the right side of a production by α', a new letter. Then $L(G')$ is the union of the words on the right sides of the productions of G'. $L(G')$ is expressible using only concatenation and union, since there are a finite number of such words. Let $P(\alpha')$ be such an expression. By Lemma 2, $L(G) = \langle P(\hat{\alpha})\rangle_\alpha$.

For the induction step suppose G has $n + 1$ variables, $\alpha_1, \ldots, \alpha_n, \sigma$, where σ is initial. For each variable β of G let G_β be the grammar with the same productions as G but with β as the initial variable. Let G_β' be the grammar derived from G_β by replacing all occurrences of σ on the right sides of productions by the new letter σ'. Let G''_β be the grammar derived from G_β' by

removing all productions with σ on the left. Then, if $\beta \neq \sigma$, $L(G''_\beta) = L(G_\beta')$ and G''_β has only n variables.

By the induction hypothesis there are expressions using only concatenation, union, and cap for $L(G'_{\alpha_i})$, $i = 1, 2, \ldots, n$. Let these be $P_i(\sigma')$. Let $Q(\alpha_1, \alpha_2, \ldots, \alpha_n, \sigma')$ be an expression for the union of the right sides of productions in G_σ' of the form $\sigma \to x$. Then

$$L(G_\sigma') = Q(P_1(\sigma'), P_2(\sigma'), \ldots, P_n(\sigma'), \sigma').$$

By Lemma 2, with a change of bound variable

$$L(G) = \langle Q(P_1(\hat{\gamma}), P_2(\hat{\gamma}), \ldots, P_n(\hat{\gamma}), \hat{\gamma})\rangle_\gamma$$

where γ is a new letter. The change of variable quaranteed that no σ-cap expression was substituted into a σ-cap expression.

COROLLARY 1. *Any notation for subfamilies of context free languages which contain all of the finite languages can be extended by the cap operator to include all context free languages.*

COROLLARY 2. *Given a context free grammar G there is an effective procedure for finding an expression for L(G).*

Proof. The proof of Theorem 1 is a constructive one.

THEOREM 2. *Any expression using the cap operator with operations under which context free languages are closed is an expression for a context free language.*

Proof. By Lemma 3 context free languages are closed under the cap operation.

COROLLARY 3. *There is an effective procedure for constructing a context free grammar for an expression using only the cap operator and other operators for which there are effective procedures.*

Proof. The proof of Lemma 3 provides the procedure for the cap operator.

It is known that if a grammar G has no self-embedding variables then $L(G)$ is regular. It is sufficient in applying the method of Theorem 1 to finding an expression for $L(G)$ to prime only those variables which are not self-embedding. In fact, not all self-embedding variables need always be primed to obtain an expression in the regular operations and cap. It may be that α is no longer self-embedding after β is primed.

If the only self-embedding variables that are primed are those that generate themselves more than once in the same word, the resulting expressions will be expressions for superlinear languages combined with the cap operator.

Some examples should make this clear. For notational purposes in the examples $G_\sigma(\alpha_1, \alpha_2, \ldots, \alpha_n)$ will signify the grammar derived from G by priming the variables $\alpha_1, \alpha_2, \ldots, \alpha_n$ on the right sides of productions, and using σ as initial variable.

EXAMPLE 1. Let G be the grammar given by the following four productions:

$$\alpha \to \lambda,$$
$$\alpha \to a\alpha b\beta,$$
$$\beta \to \lambda,$$
$$\beta \to a\alpha b\beta.$$

$L(G_\alpha)$ is the Deike language of well formed parentheses, a as left parenthesis and b as right parenthesis.

The method of the proof of Theorem 1 gives the following expressions:

$$L(G_\beta(\alpha, \beta)) = \lambda \cup a\alpha' b\beta',$$
$$L(G_\beta(\alpha)) = \langle \lambda \cup a\alpha' b\hat{\beta} \rangle_\beta,$$
$$L(G_\alpha(\alpha)) = \lambda \cup a\alpha' b\langle \lambda \cup a\alpha' b\hat{\beta} \rangle_\beta,$$
$$L(G_\alpha) = \langle \lambda \cup a\hat{\alpha} b\langle \lambda \cup a\hat{\alpha} b\hat{\beta} \rangle_\beta \rangle_\alpha.$$

The resulting expression is little more than a restatement of the original productions. A more interesting expression can be derived by starting with $G_\alpha(\alpha)$ which is a right linear grammar. Hence $L(G_\alpha(\alpha))$ is regular:

$$L(G_\alpha(\alpha)) = (a\alpha' b)^*,$$
$$L(G_\alpha) = \langle (a\hat{\alpha} b)^* \rangle_\alpha.$$

The last expression seems to give a feel for the language, with a's and b's paired, reiterated any number of times, and nested to any depth.

EXAMPLE 2. Another interesting context free language which is not superlinear was defined by Yntema (1967) in terms of matching choice sets

$$S = \bigcup_{i=0}^{\infty} S_i,$$

where $S_0 = \lambda$ and $S_{k+1} = (S_k a, bS_k)^* \circ \lambda$. S is generated by the grammar

$$\alpha \rightarrow \beta,$$

$$\beta \rightarrow \lambda,$$

$$\beta \rightarrow \alpha a \beta b \alpha.$$

Obviously $L(G_\alpha) = L(G_\beta)$, and α could be omitted if the last production were replaced by $\beta \rightarrow \beta a \beta b \beta$. Keeping the extraneous production and variable illustrates different cap expressions, however.

Starting with $G_\beta(\beta)$, which has no self-embedding elements, gives the following derivation:

$$L(G_\beta(\beta)) = \lambda \cup \beta' a \beta' b \beta',$$

$$L(G_\beta) = \langle \lambda \cup \hat{\beta} a \hat{\beta} b \hat{\beta} \rangle_\beta = S.$$

$L(G_\alpha(\alpha))$ is a superlinear language, since β is the only self-embedding variable and it cannot generate words containing itself more than once. Using matching choice notation gives a result that seems to echo the definition of S in terms of the S_i :

$$L(G_\alpha(\alpha)) = (\alpha' a, b\alpha')^* \circ \lambda,$$

$$L(G_\alpha) = \langle (\hat{\alpha} a, b\hat{\alpha})^* \circ \lambda \rangle_\alpha = S.$$

Conclusion

Cap expressions, along with other expressions for subfamilies of context free languages which include the finite languages, can be used to represent any context free language. Conversely, any language which can be expressed by the cap operator and operators under which context free languages are closed is context free. Hence, languages are context free if and only if they can be expressed in terms of the three operations union, concatenation, and cap. It is frequently convenient to include other operations, such as the regular star operation and the matching choice operation, to the list that can be used with the cap operator to characterize context free languages.

Received: April 15, 1970; Revised: December 26, 1970

References

1. Brzozowski, J. A. (1968), Regular-like expressions for some irregular languages, IEEE Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory, 278–286.
2. Ginsburg, S. and Spanier, E. (1968), Derivation-bounded languages, IEEE Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory, 306–314.
3. Yntema, M. K. (1965), A generalized regular notation used to define some families of context-free languages and to obtain their closure properties and containment relations. Ph.D. thesis, University of Illinois, Urbana, Ill.
4. Yntema, M. K. (1967), Inclusion relations among families of context-free languages, *Inform. Control* **10**, 572–597.